Informatik
Hauptcampus

HOCHSCHULE TRIER

Prozedurale Generierung von Baumstrukturen innerhalb der Unreal Engine 4

Procedural generation of tree-like structures in the Unreal Engine 4

David Liebemann

Bachelor-Abschlussarbeit

Betreuer:
Prof. Dr. Christof Rezk-Salama

Trier, 26.02.2017

Kurzfassung

Inhaltsverzeichnis

1

Einleitung und Problemstellung

test

1.1 Prozedurale Generierung

Insbesondere Vegetation - warum, was sind die Schwierigkeiten, wenn man es nicht prozedural generiert.

1.2 Bisherige Arbeiten

test

1.3 Ansatz

Verwendete Ansätze: L-Systeme mit Turtle-Graphik und Space Colonization Algorithmus. Kleine Einleitung zu beiden Ansätzen, bisherige Verwendung.

Beschreibung - was wird mit Baumstruktur / Baum-Graph gemeint?

1.4 Unreal Engine 4

Allgemeine Erklärung - was ist die Unreal Engine 4, was stellt sie zur Verfügung, wie entwickelt man dafür? Insbesondere Eingabe von Parametern über den Editor –> wichtig für Implementierung.

Erklärung - was ist ein Unreal-Akteur? Was sind die Komponenten, die diesem hinzugefügt werden können? Sämtliche Visualisierungen von L-Systemen wurden mithilfe des Unreal Engine 4 Projekts erstellt.

Erklärung - was ist ein L-System-Akteur, was ist ein Space Colonization-Akteur

Alle Längenangaben entsprechen der in der Unreal Engine 4 verwendeten Längeneinheit cm.

2

Lindenmayer-Systeme

Es werden die Regeln für die Definition von Lindenmayer-Systemen – kurz: L-Systeme – festgelegt und die verwendete Methode zur Visualisierung der Ergebnisse von L-Systemen besprochen.

2.1 Kontextfreie Grammatik

Bei L-Systemen handelt es sich um, auf Zeichenketten basierende, Ersetzungssysteme. Es werden komplexe Objekte beschrieben, indem Teile der Zeichenkette durch andere Zeichen oder Zeichenketten ersetzt werden. Die Beschreibung dieser Ersetzungen findet mittels festgelegter Produktionsregeln statt. [PL04, S.2]

Eine formale Definition eines auf Zeichenketten arbeitenden Ersetzungssystems wird durch kontextfreie Grammatiken gegeben:

Kontextfreie Grammatik: 2.1 *Eine kontextfreie Grammatik G ist ein Tupel $G = (V, N, P, \omega)$ bestehend aus:*

$\boldsymbol{V}$	*Einer nichtleeren, endlichen Menge von Buchstaben (Alphabet).*
$\boldsymbol{N}$	*Einer endlichen Menge von Variablen.*
$\boldsymbol{P}$	*Einer endlichen Menge von Produktionsregeln in der Form $P : A \to \alpha$ mit $A \in N$ und $\alpha \in (V \cup N)^*$.*
$\boldsymbol{\omega \in N}$	*Dem Axiom, Startsymbol der Grammatik.*

[Sch14, S.343]

Die Menge V^* ist die Menge aller Wörter über V – die Menge aller Wörter, die aus dem Alphabet V gebildet werden können. [Sch14, S.70]

Eine Grammatik wird als kontextfrei bezeichnet, wenn beispielsweise die Produktionsregel $A \to \alpha$ angewendet werden kann, ohne die A umgebenden Buchstaben – seinen Kontext – beachten zu müssen. [Sch14, S.343]

Eine Grammatik wird als deterministisch bezeichnet, wenn es genau eine Produktionsregel $r \in P$ für jede Variable $A \in V$ gibt, sodass $r : A \to \alpha$, $\alpha \in (V \cup N)^*$. Das bedeutet, dass die Ersetzung einer Variable eindeutig durch eine einzige Regel beschrieben wird. [STN16, S.75]

Die Anwendung der Produktionsregeln findet meist sequentiell statt – die Zeichenkette wird von links nach rechts durchlaufen und Ersetzungen werden direkt auf die untersuchte Zeichenkette angewendet. [STN16, S.75]

2.2 D0L-Systeme

Diese Arbeit beschränkt sich auf die Behandlung deterministischer, kontextfreier L-Systeme, auch D0L-Systeme genannt. Diese besitzen die Eigenschaften einer deterministischen und kontextfreien Grammatik, Produktionsregeln werden jedoch parallel und gleichzeitig auf alle Buchstaben des untersuchten Wortes angewendet. Dieses Vorgehen soll die Zellteilung in mehrzelligen Organismen simulieren und ist somit an biologische Vorgänge angelehnt. [PL04, S. 3]

Ein D0L-System kann wie folgt definiert werden:

D0L-System: 2.1 *Ein D0L-System ist ein Tupel $G = (V, P, \omega)$, bestehend aus:*

V *Einem nichtleeren, endlichen Alphabet.*

P *Einer endlichen Menge von Produktionsregeln in der Form $P : a \to b$ mit $a \in V$ und $b \in V^*$. a wird als Vorgänger, b als Nachfolger bezeichnet. Ist für einen Buchstaben $x \in V$ keine explizite Produktionsregel angegeben, wird die Identitätsproduktion $P : x \to x$ angenommen – der Buchstabe wird durch sich selbst ersetzt.*

***ω* ∈ *V*⁺** *Dem Axiom, Startsymbol der Grammatik.*

[PL04, S.4]

Die Menge V^+ ist die Menge aller nichtleeren Wörter über V.

Die Ableitung eines Wortes entspricht der Ersetzung aller Buchstaben anhand der Produktionsregeln. Ein Wort kann mehrmals abgeleitet werden.

Ableitung: 2.1 *Gegeben sei ein Wort $w = a_1...a_m$ mit $w \in V^*$ und $a_i \in V^*$. Das Wort $v = b_1...b_m$ mit $v \in V^*$ und $b_i \in V^*$ ist die Ableitung von w wenn für alle $i = 1...m$ eine Produktionsregel $a_i \to b_i$ existiert. Die Ableitung wird als $w \Rightarrow v$ notiert.*
Das Wort w_n ist die n-te Ableitung des Wortes w_0 wenn eine Folge von Wörtern $w_0, w_1, ..., w_n$ mit Ableitungen $w_0 \Rightarrow w_1 \Rightarrow ... \Rightarrow w_n$ existiert. [PL04, S.4]

Beispiel: Das Wachstum der Blaualgen-Gattung „Anabaena“ kann durch ein L-System simuliert werden. Die Buchstaben a und b beschreiben die Größe und Teilungsbereitschaft einer Algenzelle, während die Indizes l und r die Polarität einer Zelle darstellen. Es gelten folgende Produktionsregeln:

$$\begin{aligned} p_1 &: a_r \rightarrow a_l b_r \\ p_2 &: a_l \rightarrow b_l a_r \\ p_3 &: b_r \rightarrow \ a_r \\ p_4 &: b_l \rightarrow \ a_l \end{aligned} \tag{2.1}$$

Die Entwicklung einer Anfangszelle a_r (Axiom $\omega : a_r$) läuft daraufhin wie in Abbildung 2.1 dargestellt ab.

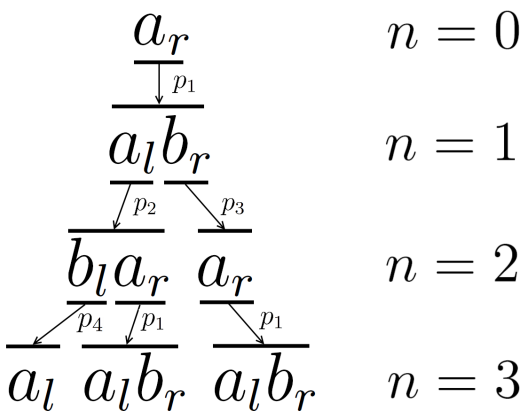


Abb. 2.1: Die n-fache Ableitung der Anfangszelle a_r anhand der Produktionsregeln $p_1 ... p_4$ aus Gleichung 2.1. Eigene Abbildung auf Basis von [PL04, S.4].

2.2.1 Parametrische L-Systeme

Parametrische L-Systeme stellen eine Erweiterung der D0L-Systeme dar. Die Buchstaben eines verwendeten Alphabets V werden um zugeordnete Parameter aus der Menge der reelen Zahlen ergänzt. Ein solches parametrisches Wort $V \times \mathbb{R}^*$ besteht aus einem Zeichen $A \in V$ und Parametern $a_1, ..., a_n \in \mathbb{R}$ und wird als $A(a_1, ..., a_n)$ dargestellt. Ein parametrisches Wort ohne Parameter mit dem Zeichen $A \in V$ wird schlicht als A dargestellt. [PL04, S.41]

Während in der Definition von Modulen numerische Konstanten verwendet werden, werden bei der Angabe eines L-Systems Variablen verwendet. Ist Σ eine Menge von Variablen, dann ist $E(\Sigma)$ ein arithmetischer Ausdruck, in dem Variablen, Konstanten und arithmetische Operatoren auf eine zulässige Weise kombiniert werden. [PL04, S.41]

Parametrisches L-System: 2.2.1 *Ein Parametrisches L-System ist ein Tupel* $G = (V, \Sigma, P, \omega)$*, bestehend aus:*

V *Einem nichtleeren, endlichen Alphabet.*

Σ *Einer Menge von Variablen.*

$\boldsymbol{P}$ *Einer endlichen Menge von Produktionsregeln* $P : (V \times \Sigma^*) \rightarrow (V \times E(\Sigma))^*$

$\boldsymbol{\omega \in M^+}$ *mit* $M = (V \times \mathbb{R}^*)$ *– einem Axiom in Form eines nichtleeren, parametrischen Wortes.*

[PL04, S.41]

Eine Produktion kann auf ein parametrisches Wort angewendet werden wenn das Zeichen, das dem Wort vorausgeht, und die Anzahl der Parameter mit dem Zeichen und der Anzahl der Parameter im Vorgänger der Produktionsregel übereinstimmen. [PL04, S.42]

Beispiel: Gegeben sei folgendes, parametrisches L-System:

$$\begin{aligned} \omega &: A(1,1) \\ p_1 &: A(x,y) \rightarrow A(x+1, y*2)\ B(y) \\ p_2 &: B(x) \quad \rightarrow B(x+1)\ C \end{aligned} \tag{2.2}$$

Das Alphabet V und die Menge der Variablen Σ gehen implizit aus der Angabe der Produktionsregeln hervor und werden in zukünftigen L-System-Gleichungen nicht angegeben. Die Entwicklung des L-Systems läuft anhand Abbildung 2.2 ab.

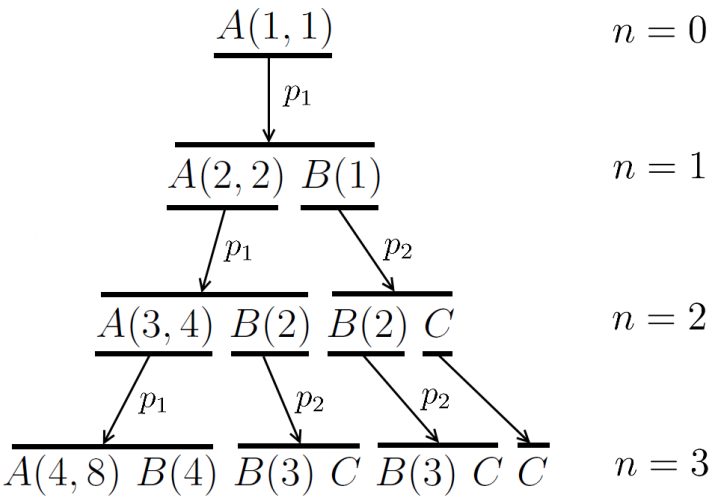


Abb. 2.2: Die n-fache Ableitung des Axioms $A(1,1)$ anhand der Produktionsregeln aus Gleichung 2.2. Die Ersetzung des Zeichens C erfolgt anhand der impliziten Identitätsproduktion. Eigene Abbildung.

2.3 Grafische Interpretation von L-Systemen

Um die Ergebnisse von L-Systemen in Form von dreidimensionalen Objekten zu visualisieren, muss eine grafische Interpretation der resultierenden Zeichenket-

ten festgelegt werden. Im Folgenden wird die verwendete Visualisierungsmethode, Turtle-Interpretation, vorgestellt.

2.3.1 Turtle-Interpretation

Die Turtle-Interpretation im zweidimensionalen Raum entspricht der Vorstellung einer Turtle[1] auf einem Blatt Papier. Die Turtle besitzt eine Position $\overrightarrow{p}$ sowie einen Einheitsvektor $\overrightarrow{H}$ in kartesischen Koordinaten. $\overrightarrow{p}$ beschreibt die Position der Turtle auf der Ebene und $\overrightarrow{H}$ entspricht der Blickrichtung (Heading) der Turtle. Der Zustand einer Turtle wird somit vollständig durch die Position und Blickrichtung definiert und wird als Tupel $(\overrightarrow{p}, \overrightarrow{H})$ angegeben. [GSJ04, S.2] Die Ausgangsposition entspricht dem Ursprung des lokalen Koordinatensystems der Turtle.

Es können drei Aktionen durchführt werden, welche durch die folgenden Symbole dargestellt werden:

***F*(*l*)** Die Turtle bewegt sich um $l > 0$ in die Richtung der aktuellen Blickrichtung. Die neue Position ist $\overrightarrow{p_{neu}}$ mit:

$$\overrightarrow{p_{neu}} = \overrightarrow{p} + \overrightarrow{H} * l \tag{2.3}$$

Zwischen der alten Position $\overrightarrow{p}$ und der neuen Position $\overrightarrow{p_{neu}}$ wird eine Linie gezeichnet.

+(*d*) Die Turtle dreht sich um den Winkel d nach links. Die neue Blickrichtung ist $\overrightarrow{H'}$ mit:

$$\overrightarrow{H'} = \begin{pmatrix} cos(d) & -sin(d) \\ sin(d) & cos(d) \end{pmatrix} * \overrightarrow{H} \tag{2.4}$$

−(*d*) Die Turtle dreht sich um den Winkel d nach rechts. Die neue Blickrichtung ist $\overrightarrow{H'}$ mit:

$$\overrightarrow{H'} = \begin{pmatrix} cos(d) & sin(d) \\ -sin(d) & cos(d) \end{pmatrix} * \overrightarrow{H} \tag{2.5}$$

[GSJ04, S.4,46] [PL04, S.7] Die Symbole „+“ und „-“ werden sowohl im Alphabet eines L-Systems als auch bei arithmetischen Operationen in Parameterangaben verwendet, ihre Bedeutung ist abhängig vom Kontext, in dem sie angewendet werden. [PL04, S.46]

Die grafische Turtle-Interpretation einer Zeichenkette, die durch ein L-System zurückgegeben wird, sind somit die Linien, die auf Grundlage der definierten Symbole gezeichnet werden.

Beispiel: Mithilfe von L-Systemen und einer Turtle-Interpretation können Fraktale, in diesem Beispiel sogenannte Koch-Kurven, visualisiert werden. Diese Kurven bestehen aus einem Initiator – einer einfachen, zweidimensionalen Form – und

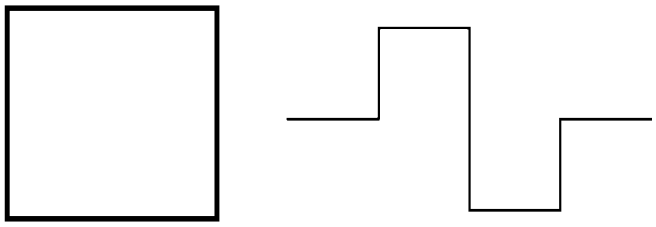


Abb. 2.3: Links: Initiator der Koch-Kurve in Form eines einfachen Quadrats. Rechts: Generator der Koch-Kurve in Form eines offenen Polygonzugs. Eigene Abbildungen.

einem Generator, der einem offenen Polygonzug entspricht. In jedem Ableitungsschritt, angefangen bei dem Initiator, wird jede gerade Linie durch den Generator ersetzt. [Man83, S.39]

Dieses Verhalten kann nun auf ein L-System abgebildet werden: Der Initiator entspricht dem Axiom und der Generator einer Produktionsregel des L-Systems. Der in Abbildung 2.3 dargestellte Initiator und Generator entsprechen der Turtle-Interpretation des folgenden L-Systems:

$$\begin{aligned} \omega &: F - F - F - F \\ p_1 &: F \rightarrow F + F - F - FF + F + F - F \end{aligned} \tag{2.6}$$

Für eine bessere Übersicht wurde die Angabe der Parameter weggelassen. Die Turtle interpretiert F als $F(l)$, $-$ als $-(d)$ und $+$ als $+(d)$ mit festgelegter Strichlänge l und Drehwinkel d. Die Entwicklung des L-Systems läuft, als Turtle-Interpretation visualisiert, wie in Abbildung 2.4 dargestellt ab.

2.3.2 Verzweigte L-Systeme

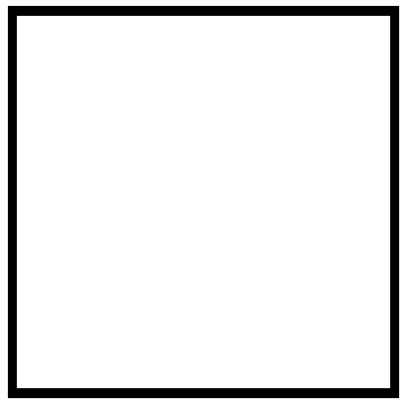
Die bisherigen Definitionen von L-Systemen und die korrespondierende Turtle-Interpretation lässt lediglich die Bildung von Grafiken zu, die einen einzelnen, zusammenhängenden Polygonzug erlauben. Um L-Systeme zu bilden, deren Visualisierungen Baumstrukturen ähneln muss die bisherige Turtle-Interpretation um die Möglichkeit erweitert werden, Verzweigungen zu verarbeiten. [PL04, S.24]

Folgende Operationen werden eingeführt:

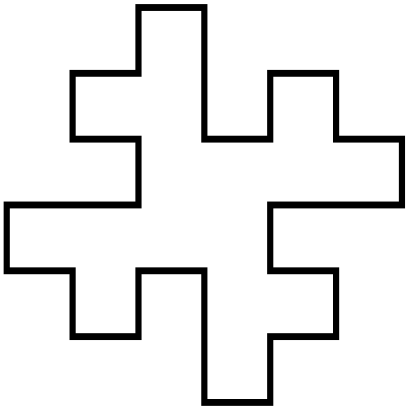
[Der aktuelle Zustand der Turtle in Form ihrer Position und Rotation wird auf einem Kellerspeicher (Stack) abgelegt.

] Der oberste Zustand der Turtle wird vom Kellerspeicher genommen. Die aktuelle Position und Rotation der Turtle wird auf die im Zustand gespeicherte Position und Rotation gesetzt. Es wird keine Linie zwischen der alten und neuen Position gezeichnet.

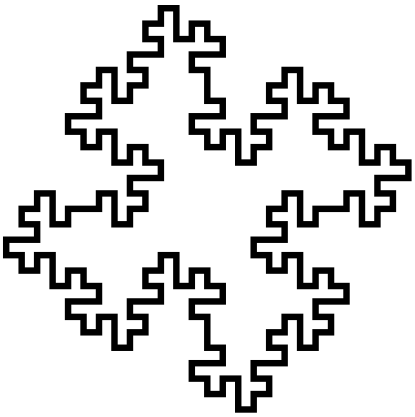
[1] „Turtle“ – englisch für „Schildkröte“.



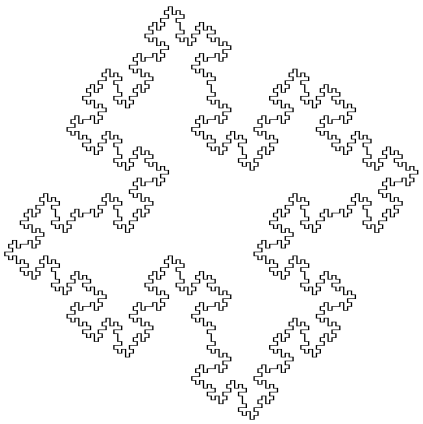
(a) $n = 0$, $l = 400$, $d = 90°$



(b) $n = 1$, $l = 100$, $d = 90°$



(c) $n = 2$, $l = 25$, $d = 90°$



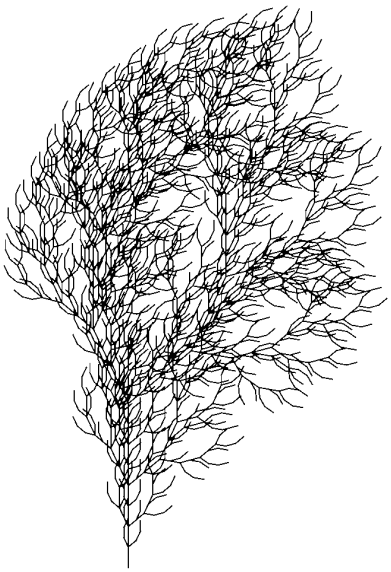
(d) $n = 3$, $l = 6.25$, $d = 90°$

Abb. 2.4: Die n-fache Ableitung des Axioms ω anhand der Produktionsregel p_1 aus Gleichung 2.6, visualisiert mithilfe der implementierten Turtle-Interpretation. Eigene Abbildungen.

[PL04, S.24]

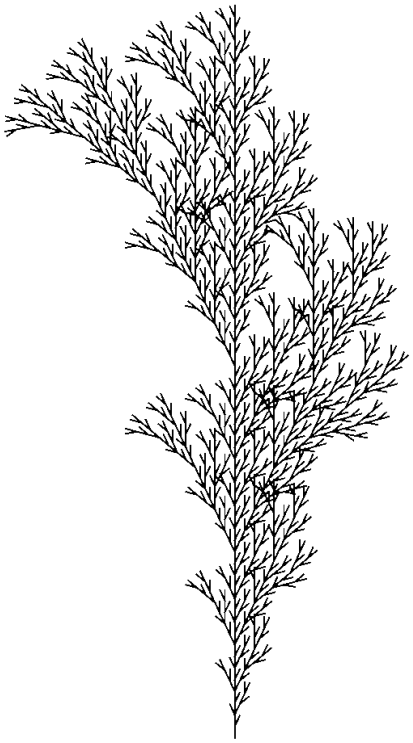
Diese Erweiterung erlaubt es mehrere Linien zu zeichnen, die von einem einzigen Punkt ausgehen – die Visualisierung von Abzweigungen ist somit möglich. Die Operatoren **[** und **]** markieren den Anfang und das Ende eines Zweiges. [PL04, S.24]

Beispiel: Mithilfe von verzweigten L-System-Beschreibungen lassen sich die in Abbildung 2.5 gezeigten Strukturen bilden. Die Turtle-Interpretation folgt der in 2.3.1 beschriebenen Interpretation mit fester Strichlänge l und festem Drehwinkel d.



(a) $n = 4$, $l = 18$, $d = 25°$

$\omega \ : F$
$p_1 : F \rightarrow FF - [-F + F + F] + [+F - F - F]$
[PL04, S.25]



(b) $n = 5$, $l = 15$, $d = 25°$

$\omega \ : F$
$p_1 : F \rightarrow F[-F]F[+F][F]$
[STN16, S.78]

Abb. 2.5: Die n-fache Ableitung der Axiome anhand der Produktionsregeln, visualisiert mithilfe der implementierten Turtle-Interpretation. Eigene Abbildung.

2.3.3 Erweiterung der Turtle-Interpretation in den dreidimensionalen Raum

Die bisherige Definition eines Turtle-Zustands mithilfe einer zweidimensionalen Position und einer Blickrichtung genügt nicht, um Visualisierungen von L-Systemen in Form von dreidimensionalen Baumstrukturen zu ermöglichen. Sowohl die Zustands-Definition als auch die interpretierten Operationen müssen angepasst und erweitert werden.

Der Zustand der Turtle im dreidimensionalen Raum besitzt eine Position $\overrightarrow{p}$ sowie eine 3×3 Rotationsmatrix $\boldsymbol{R}$, welche die Orientierung der Turtle im Raum beschreibt. Der Zustand wird als Tupel $(\overrightarrow{p}, \boldsymbol{R})$ angegeben. $\boldsymbol{R}$ entspricht zu Anfang einer Identitätsmatrix. Die Einheitsvektoren H, L und U sind orthogonal zueinander und bilden das lokale Koordinatensystem der Turtle.

$\overrightarrow{\boldsymbol{H}}$ Die Blickrichtung (Heading-Vektor) der Turtle. Eine Rotation um diesen Vektor um den Winkel d entspricht der Rotationsmatrix $R_{\overrightarrow{H}}$:

$$R_{\overrightarrow{H}}(d) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(d) & sin(d) \\ 0 & -sin(d) & cos(d) \end{pmatrix} \tag{2.7}$$

$\overrightarrow{\boldsymbol{L}}$ Der Vektor, der im lokalen Koordinatensystem der Turtle nach links zeigt (Left-Vektor). Eine Rotation um diesen Vektor um den Winkel d entspricht der Rotationsmatrix $R_{\overrightarrow{L}}$:

$$R_{\overrightarrow{L}}(d) = \begin{pmatrix} cos(d) & 0 & sin(d) \\ 0 & 1 & 0 \\ -sin(d) & 0 & cos(d) \end{pmatrix} \tag{2.8}$$

$\overrightarrow{\boldsymbol{U}}$ Der Vektor, der im lokalen Koordinatensystem der Turtle nach oben zeigt (Up-Vektor). Eine Rotation um diesen Vektor um den Winkel d entspricht der Rotationsmatrix $R_{\overrightarrow{U}}$:

$$R_{\overrightarrow{U}}(d) = \begin{pmatrix} cos(d) & -sin(d) & 0 \\ sin(d) & cos(d) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.9}$$

[PL04, S.19] [DL05, S.69]

Die erweiterte Turtle-Interpretation verarbeitet folgende Symbole:

$\boldsymbol{F(l)}$ Die Turtle bewegt sich um $l > 0$ in Blickrichtung $\overrightarrow{H}$. Die neue Position ist $\overrightarrow{p_{neu}}$ mit:

$$\overrightarrow{p_{neu}} = \overrightarrow{p} + \boldsymbol{R} * \overrightarrow{H} * l \tag{2.10}$$

Zwischen der alten Position p und der neuen Position p_{neu} wird eine Linie gezeichnet.

$\boldsymbol{+(d)}$ Die Turtle dreht sich nach links um den Winkel d. Die neue Rotationsmatrix entspricht $\boldsymbol{R'}$ mit:

$$\boldsymbol{R'} = \boldsymbol{R} * R_{\overrightarrow{U}}(d) \tag{2.11}$$

$\boldsymbol{-(d)}$ Die Turtle dreht sich nach rechts um den Winkel d. Die neue Rotationsmatrix entspricht $\boldsymbol{R'}$ mit:

$$\boldsymbol{R'} = \boldsymbol{R} * R_{\overrightarrow{U}}(-d) \tag{2.12}$$

$\boldsymbol{\&(d)}$ Die Turtle neigt sich nach unten um den Winkel d. Die neue Rotationsmatrix entspricht $\boldsymbol{R'}$ mit:

$$\boldsymbol{R'} = \boldsymbol{R} * R_{\overrightarrow{L}}(d) \tag{2.13}$$

$\boldsymbol{\hat{}(d)}$ Die Turtle neigt sich nach oben um den Winkel d. Die neue Rotationsmatrix entspricht $\boldsymbol{R'}$ mit:

$$\boldsymbol{R'} = \boldsymbol{R} * R_{\overrightarrow{L}}(-d) \tag{2.14}$$

\(d) Die Turtle rollt sich nach links um den Winkel d. Die neue Rotationsmatrix entspricht $\boldsymbol{R'}$ mit:

$$\boldsymbol{R'} = \boldsymbol{R} * R_{\overrightarrow{H}}(d) \tag{2.15}$$

/(d) Die Turtle rollt sich nach rechts um den Winkel d. Die neue Rotationsmatrix entspricht $\boldsymbol{R'}$ mit:

$$\boldsymbol{R'} = \boldsymbol{R} * R_{\overrightarrow{H}}(-d) \tag{2.16}$$

[PL04, S.19] [DL05, S.69]

Mithilfe der Turtle-Interpretation im dreidimensionalen Raum können nun L-Systeme visualisiert werden, die Baumstrukturen ähneln. Ein Beispiel dafür ist das L-System aus Gleichung 2.17, dargestellt in Abbildung 2.6.

$$\begin{array}{ll} \omega: & /(45)\ A \\ p_1: A & \rightarrow F(50)\ [\&(a)F(50)A]\ /(d)\ [\&(a)F(50)A]\ /(d)\ [\&(a)F(50)A] \\ p_2: F(l) & \rightarrow F(l * l_r) \end{array} \tag{2.17}$$

[PL04, S.60]

2.4 Anpassungen für Baumstrukturen

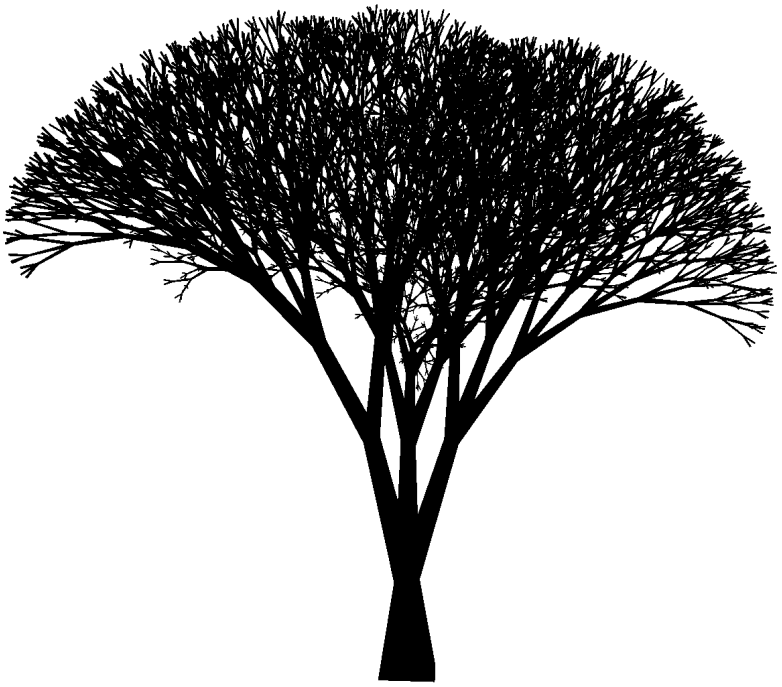
Die Darstellung von verzweigten L-Systemen mithilfe von Zeichenketten ermöglicht eine Repräsentation in Textform und erlaubt die einfache Definition von Produktionsregeln. Um jedoch eine effiziente Nachbearbeitung und Visualisierung der Ergebnisse eines L-Systems zu ermöglichen, müssen die Aktionen einer Turtle als graphentheoretischer Baum festgehalten werden. Weiterhin wird der Begriff des Tropismus und dessen Einfluss auf die Modellierung von Baumstrukturen eingeführt.

2.4.1 Graphentheoretische Bäume

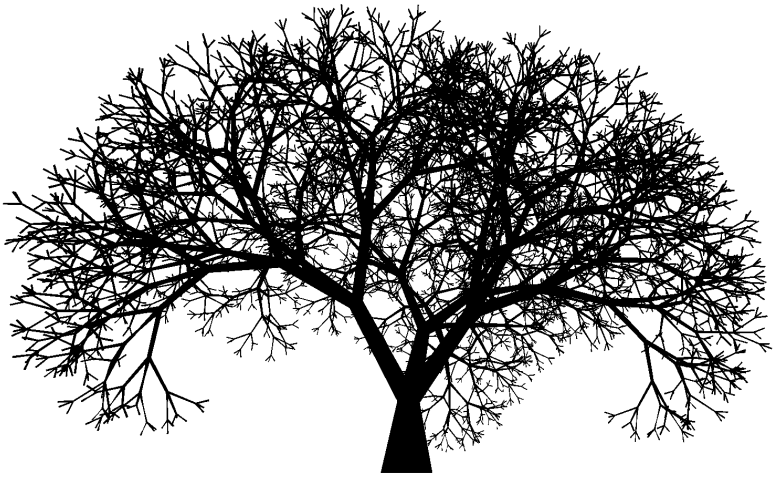
Ein Baum, in der Graphentheorie, ist ein kreisfreier Graph $G = \langle V, E \rangle$ für den folgende Begriffe festgelegt werden:

V Die Menge der Knoten. Jeder Knoten entspricht einem Punkt $\overrightarrow{p_v}$ im dreidimensionalen Raum. [Sch14, S.358]

E Die Menge der Kanten, welche die Vorgänger-Nachfolger-Beziehungen zwischen den Knoten darstellt. Eine Kante $e \in E$ wird als Tupel $e = (v_1, v_2)$ mit $v_1, v_2 \in V$ dargestellt. v_1 wird als Vorgänger von v_2 und v_2 als Nachfolger von v_1 bezeichnet. Jeder Knoten besitzt maximal einen Vorgänger und eine endliche Menge von Nachfolgern. [Sch14, S.358] [Lux14, S.29]



(a) $n = 8$, $d = 137.5°$, $a = 18.95°$, $l_r = 1.3$



(b) $n = 8$, $d = 137.5°$, $a = 36.0°$, $l_r = 1.3$

Abb. 2.6: Die n-fache Ableitung des Axioms anhand der Produktionsregeln aus Gleichung 2.17, visualisiert mithilfe der implementierten Turtle-Interpretation. Eigene Abbildungen.

Wurzel Eine Wurzel ist ein Knoten $v \in V$ ohne einen Vorgänger. Der Baum besitzt genau eine Wurzel. [Sch14, S.358]

Grad Der Grad eines Knotens ist definiert als die Anzahl seiner Nachfolger. [Lux14, S.29]

Tiefe Die Tiefe eines Knotens ist definiert als die Länge der Folge von Vorgängern, die durchlaufen werden muss, bis die Wurzel erreicht wurde. Die Wurzel besitzt die Tiefe 0. [Lux14, S.30]

Ein Baum ist grundsätzlich ein rein topologisches Objekt, mithilfe der Zuordnung von Punkten zu Knoten kann jedoch eine geometrische Vorstellung des Aufbaus bewirkt werden. [PL04, S.23]

Um den Aufbau eines Baumes $G = \langle V, E \rangle$ mithilfe der Turtle-Interpretation zu ermöglichen, muss der Turtle-Zustand um einen Knotenpunkt $v \in V$ ergänzt werden. Der Zustand wird als Tupel $\overrightarrow{p}, v, \boldsymbol{R}$ angegeben.

Die Turtle-Interpretation beginnt mit der Erstellung der Wurzel am Ursprung des lokalen Bezugssystems. Die Verarbeitung folgender Symbole muss erweitert werden:

F(l) Die neue Position $\overrightarrow{p_{neu}}$ der Turtle wird entsprechend Gleichung 2.10 berechnet. An diesem Punkt wird ein Knoten v_{neu} und eine Kante (v, v_{neu}) erstellt sowie v als Vorgänger von v_{neu} und v_{neu} als Nachfolger von v eingetragen. v entspricht dem aktuellen Zustandsknoten der Turtle. Der neue Turtle-Zustand ist $(\overrightarrow{p_{neu}}, v_{neu}, \boldsymbol{R})$.

[Zusätzlich zu der Position und Rotation der Turtle wird auch der aktuelle Knoten auf einem Kellerspeicher abgelegt.

] Der oberste Zustand der Turtle wird vom Kellerspeicher genommen. Zusätzlich zur Position und Rotation wird auch der Zustandsknoten der Turtle auf den gespeicherten Knoten gesetzt.

Die Kanten zwischen Knoten wurden durch die Turtle-Interpretation bisher als Linien visualisiert und können als Astsegmente von echten Baumstrukturen verstanden werden. [PL04, S.23] Die Repräsentation von Turtle-Aktionen als graphentheoretischer Baum ermöglicht jedoch die in Abschnitt 4.4 beschriebene Generierung und Nachbearbeitung von Modelldaten.

2.4.2 Tropismus

Tropismus ist die Tendenz einer Pflanze in eine bestimmte Richtung zu wachsen, beispielsweise aufgrund einer Lichtquelle oder der Beugung durch Gravitation.[RLP07, Abschn. 3] Der Einfluss von Tropismus wird als ein dreidimensionaler Vektor $\overrightarrow{T}$ angegeben. Die Bewegung $F(l)$ der Turtle wird wie folgt erweitert:

***F*(*l*)** Die Turtle bewegt sich um $l > 0$ in Blickrichtung $\overrightarrow{H}$. Die neue Position ist $\overrightarrow{p_{neu}}$ mit:

$$\overrightarrow{p_{neu}} = \overrightarrow{p} + l * \frac{\overrightarrow{H_{rot}} + e * \overrightarrow{T}}{\|\overrightarrow{H_{rot}} + e * \overrightarrow{T}\|} \tag{2.18}$$

$$\text{und } \overrightarrow{H_{rot}} = \boldsymbol{R} * \overrightarrow{H} \tag{2.19}$$

wobei e der Anfälligkeit des Baums für die Beugung durch Tropismus entspricht. [PL04, S.58] [RLP07, Abschn. 3]

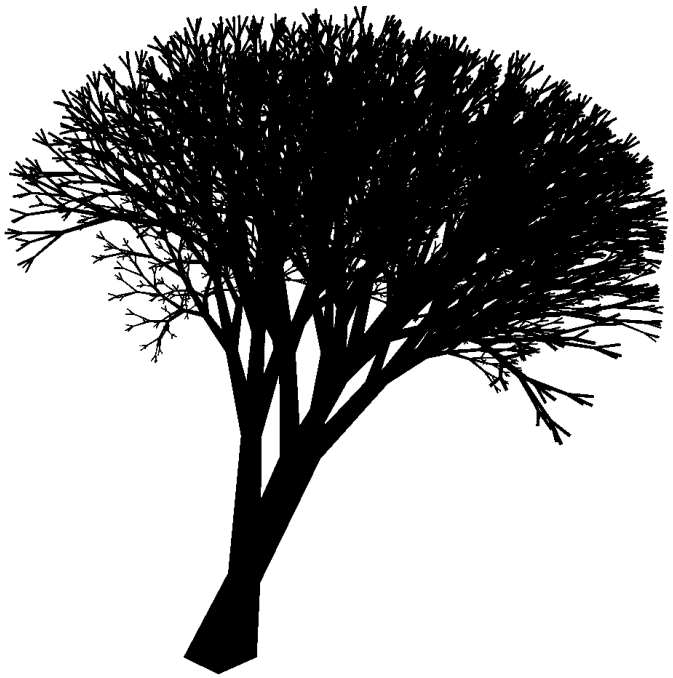


Abb. 2.7: .
$n = 8$, $d = 137.5°$, $a = 36.0°$, $l_r = 1.3$, $\overrightarrow{T} = \begin{pmatrix} 0 \\ 1 \\ -0.5 \end{pmatrix}$, $e = 0.27$
Die n-fache Ableitung des Axioms anhand der Produktionsregeln aus Gleichung 2.17, visualisiert mithilfe der erweiterten Turtle-Interpretation. Zeigt den Einfluss von Tropismus auf den in Abbildung 2.6b dargestellten Baum. Eigene Abbildung.

3

Space Colonization Algorithmus

Es werden die benötigten Eingaben und der Ablauf des Space Colonization Algorithmus festgelegt. Die Prozedur zur Generierung von Baumstrukturen wird vorgestellt und die implementierten Erweiterungen des ursprünglichen Algorithmus werden erläutert.

3.1 Ursprung

Der Space Colonization Algorithmus wurde ursprünglich zur Modellierung und Visualisierung von Blattvenen vorgeschlagen und basiert auf der Wirkung des Pflanzenhormons Auxin. Dieser Hormonstoff entsteht im Blatt und wird von bereits existierenden Blattvenen angezogen, der resultierende Hormonstrom führt zur Bildung von neuen Venen im Blatt. Die Simulation dieses Vorgangs führt zu realitätsnahen Venenmustern. [RFL+05, Abschn. 2.5]

Mithilfe einer Erweiterung in den dreidimensionalen Raum und Nachbearbeitung der Resultate kann eine Vielfalt von Baum- und Strauchstrukturen generiert werden. [RLP07, Abschn. 1]

3.2 Aufbau

Der Algorithmus verarbeitet eine Menge von Einflusspunkten S und baut darauf basierend einen Baum $G = \langle V, E \rangle$ auf.

Der Algorithmus benötigt die folgenden Eingaben:

d_i Der Einflussradius. Einflusspunkte prägen den Aufbau des Baums nur, wenn sich Knotenpunkte innerhalb dieses Radius befinden. [RLP07, Abschn. 2]

d_k Der Minimalradius. Befindet sich ein Knotenpunkt innerhalb des Minimalradius um einen Einflusspunkt, wird dieser aus der Menge der Einflusspunkte S entfernt. [RLP07, Abschn. 2]

D Die Schrittweite. Jeder neu generierte Knotenpunkt wird in diesem Abstand zu seinem Vorgänger positioniert. [RLP07, Abschn. 2]

$\overrightarrow{\boldsymbol{T}}$ Der Tropismusvektor.

3.3 Ablauf

Zu Beginn des Algorithmus werden N Einflusspunkte in einem vorgegebenen Bereich generiert. Dieser Einflussbereich signalisiert die Verfügbarkeit von Raum, in dem die Baumstruktur entstehen kann. [RLP07, Abschn. 2]

Daraufhin wird der Baum iterativ aufgebaut und durchläuft in jeder Iteration die folgenden Schritte:

1. Für jeden Einflusspunkt in S wird der am nächsten liegende Knotenpunkt $v \in V$ bestimmt. Befindet sich v innerhalb des Einflussradius d_i um den Einflusspunkt herum, wird dieser einer zugeordneten Menge $S(v)$ hinzugefügt. $S(v)$ beinhaltet somit alle Einflusspunkte, die einen Einfluss auf den Knotenpunkt ausüben. [RLP07, Abschn. 2]

2. Befinden sich Elemente in $S(v)$, wird ein neuer Knotenpunkt v' den Nachfolgern von v hinzugefügt und v als Vorgänger von v' eingetragen. Alle Punkte in $S(v)$ beeinflussen v' in gleichem Maße, die neue Position $\overrightarrow{p_{v'}}$ des Knotenpunkts berechnet sich somit wie folgt:

$$\overrightarrow{p_{v'}} = \overrightarrow{p_v} + D * \tilde{n} \tag{3.1}$$

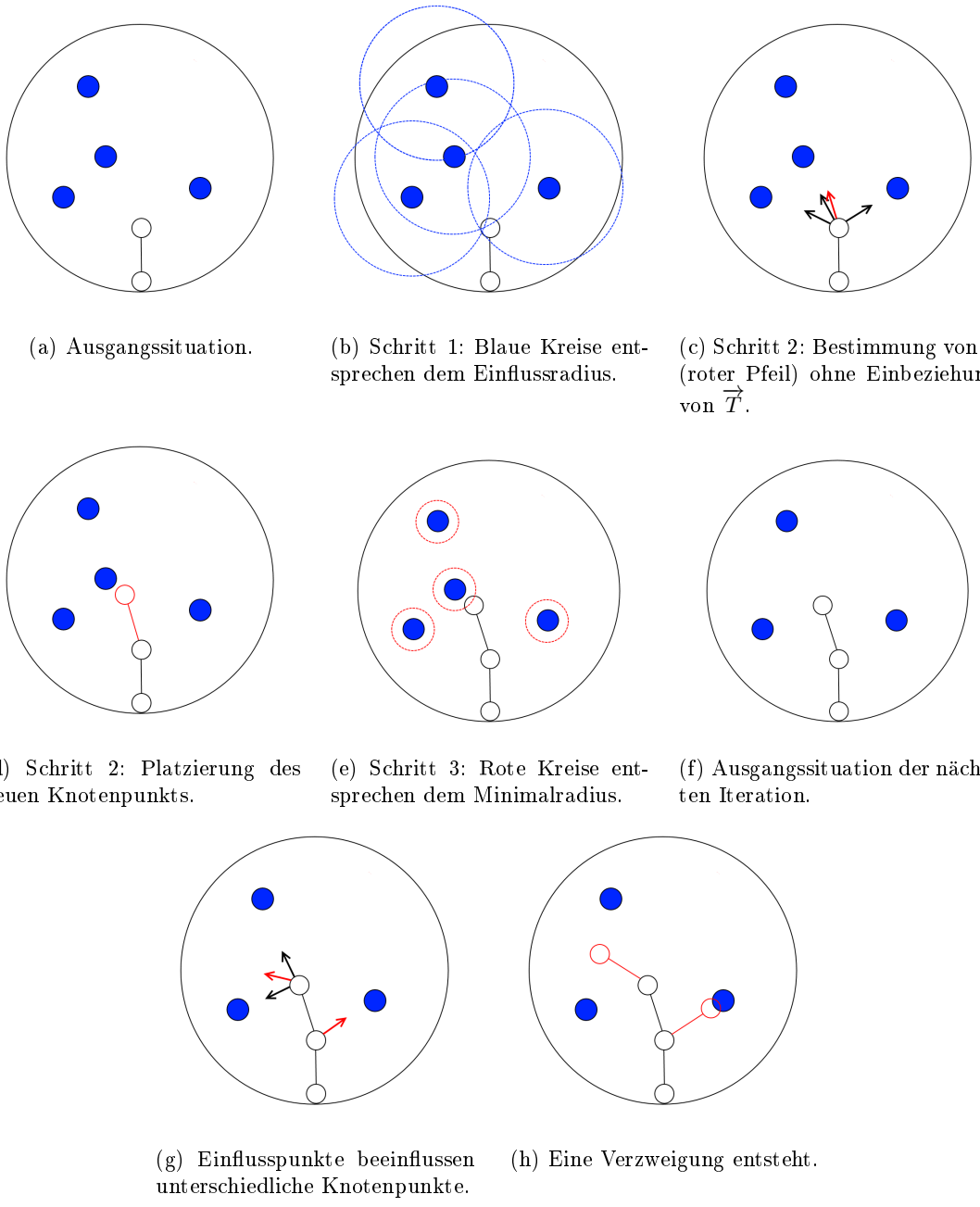
$$\text{mit } \tilde{n} = \frac{\overrightarrow{n} + \overrightarrow{T}}{\|\overrightarrow{n} + \overrightarrow{T}\|} \tag{3.2}$$

$$\text{und } \overrightarrow{n} = \sum_{s \in S(v)} \frac{\overrightarrow{p_s} - \overrightarrow{p_v}}{\|\overrightarrow{p_s} - \overrightarrow{p_v}\|} \tag{3.3}$$

[RLP07, Abschn. 2]

3. Für jeden Einflusspunkt wird überprüft, ob sich ein Knotenpunkt innerhalb des Minimalradius d_k befindet. Existiert ein solcher Knotenpunkt, wird der Einflusspunkt aus der Menge der Einflusspunkte S entfernt. [RLP07, Abschn. 2]

Diese Schritte werden solange ausgeführt, bis alle Einflusspunkte entfernt wurden, sich kein Knotenpunkt im Einflussradius eines Einflusspunktes befindet oder bis eine vorgegebene Maximalanzahl von Iterationen durchgeführt wurde. Abbildung 3.1 zeigt eine beispielhafte Anwendung des Algorithmus.



(a) Ausgangssituation.

(b) Schritt 1: Blaue Kreise entsprechen dem Einflussradius.

(c) Schritt 2: Bestimmung von $\tilde{n}$ (roter Pfeil) ohne Einbeziehung von $\vec{T}$.

(d) Schritt 2: Platzierung des neuen Knotenpunkts.

(e) Schritt 3: Rote Kreise entsprechen dem Minimalradius.

(f) Ausgangssituation der nächsten Iteration.

(g) Einflusspunkte beeinflussen unterschiedliche Knotenpunkte.

(h) Eine Verzweigung entsteht.

Abb. 3.1: Beispielhafte Anwendung des Space Colonization Algorithmus. Blaue Punkte entsprechen Einflusspunkten, weiße Punkte entsprechen Knotenpunkten. Der unterste Knotenpunkt stellt die Wurzel dar. Eigene Abbildungen auf Grundlage von [RLP07, Abb. 2].

3.4 Generierung von Baumstrukturen

Der Space Colonization Algorithmus liefert einen Baum, der Knotenpunkte enthält, welche Positionen im dreidimensionalen Raum darstellen. Um diese Knotenpunkte in Form von baumähnlichen Strukturen zu visualisieren, wird die Prozedur erweitert. Der Aufbau der Baumstruktur läuft wie folgt ab:

1. Der Einflussbereich wird mit der vorgegeben Anzahl von Einflusspunkten gefüllt. [RLP07, Abschn. 2]

2. Der Baum wird wie in Abschnitt 3.3 beschrieben iterativ generiert. [RLP07, Abschn. 2]

3. Die Nachfolger jedes Knotenpunkts werden einander angenähert, um eine Verringerung der Abzweigungswinkel zwischen den verbindenden Kanten zu erreichen. Dies führt zu einer insgesamt realistischeren Baumstruktur. [RLP07, Abschn. 2]

4. Die Kanten, welche die Knotenpunkte verbinden, werden mithilfe von Zylindern visualisiert, um die Aststruktur eines biologischen Baumes zu simulieren. [RLP07, Abschn. 2]

Die von Runions u.a. [RLP07] vorgeschlagene Kurven-Unterteilung [RLP07, Abschn. 2] wurde in dieser Arbeit nicht behandelt, da durch Angabe der Schrittweite D eine ausreichende visuelle Qualität erzielt wurde.

Abbildung 3.2 zeigt die Modellierung einer zweidimensionalen Baumstruktur.

Verringerung der Abzweiungswinkel

Die m Nachfolger $v_1 ... v_m$ eines Knotens v werden wie folgt einander angenähert:

$$\overrightarrow{m} = \overrightarrow{p_v} + \frac{\sum_{i=0}^{m} (\overrightarrow{p_{v_i}} - \overrightarrow{p_v})}{m} \tag{3.4}$$

wobei $\overrightarrow{m}$ dem arithmetischen Mittelpunkt der Nachfolger entspricht. Die Positionen der Nachfolger werden nun wie folgt dem Mittelpunkt angenähert:

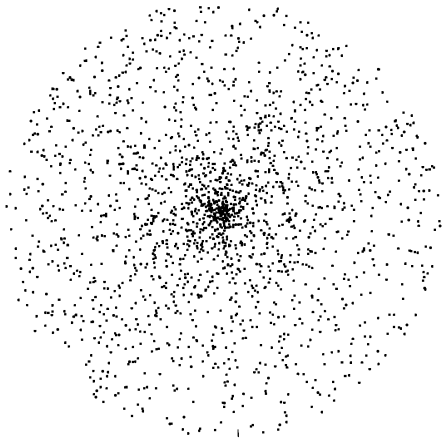
$$\overrightarrow{p_{v_i}} = \overrightarrow{p_{v_i}} + \frac{\overrightarrow{m} - \overrightarrow{p_{v_i}}}{2} \tag{3.5}$$

Berechnung der Zylinderbreiten

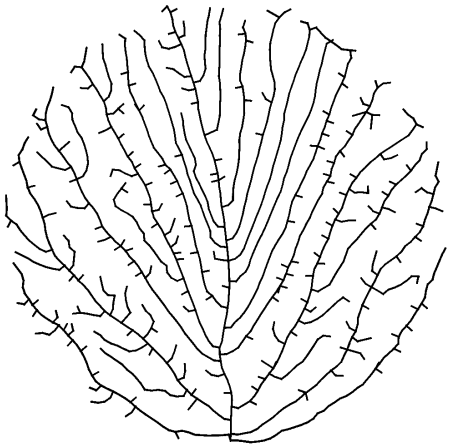
Die Berechnung der Zylinderbreiten erfolgt anhand von Murrays Regel, die besagt, dass der Radius r eines Astes auf Grundlage der Radien $r_{n_1} ... r_{n_m}$ der nachfolgenden, von ihm abzweigenden Äste wie folgt berechnet werden kann:

$$r^g = r^g_{n_1} + r^g_{n_2} + ... + r^g_{n_m} \tag{3.6}$$

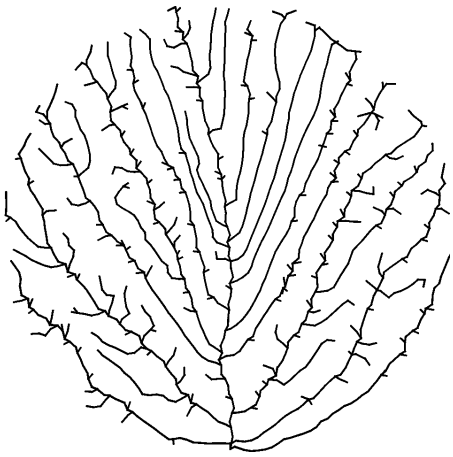
Diese Berechnung kann rekursiv auf dem Baum ausgeführt werden, indem für jeden Knotenpunkt der Radius aus den Radien seiner Nachfolger berechnet wird. Besitzt ein Knotenpunkt keine Nachfolger, wird ein festgelegter Radius r_0 zurückgegeben. [RFL+05, Abschn. 3.5] Der Wert g kann frei gewählt werden.



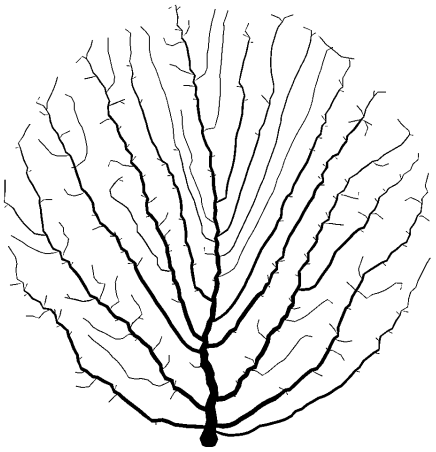
(a) $N = 2000$ Einflusspunkte, zufällig in einem Ring mit dem Radius $r = 500$ verteilt.



(b) Ergebnis des Space Colonization Algorithmus mit einem Einflussradius $d_i = 100$, Minimalradius $d_k = 20$, Schritteweite $D = 15$ und $\overrightarrow{T} = \overrightarrow{0}$.



(c) Verringerung der Abzweigungswinkel.



(d) Modellierung des Ergebnis mithilfe von Zylindern mit $r_0 = 1$ und $g = 2$.

Abb. 3.2: Modellierung einer zweidimensionalen Baumstruktur entsprechend der in Abschnitt 3.4 beschriebenen Schritte. Eigene Abbildungen.

3.5 Erweiterungen

Die folgenden Erweiterungen des ursprünglichen Algorithmus wurden implementiert.

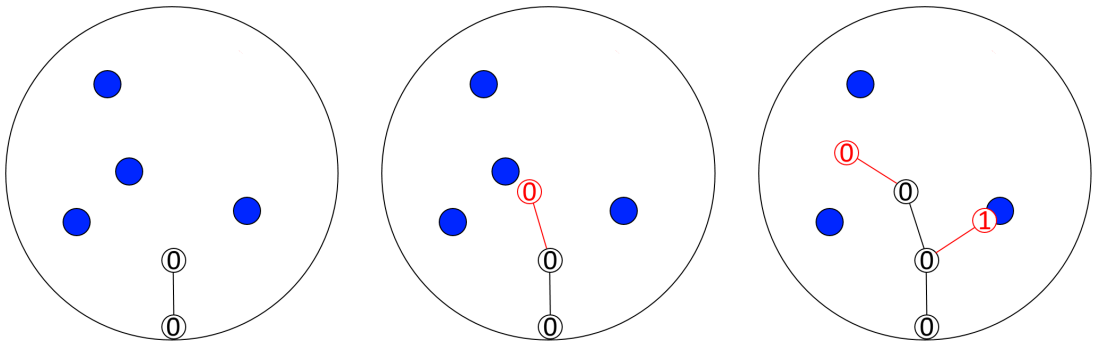
Zweigtiefe

Die Zweigtiefe $Z(v)$ eines Knotens $v \in V$ wird für Erweiterungen des Space Colonization Algorithmus und die Generierung von Modelldaten verwendet. Sie entspricht nicht der Tiefe des Knotens und berechnet sich wie folgt:

$$Z(v) = \begin{cases} 0 & \text{falls v die Wurzel ist} \\ Z(v') & \text{falls v' genau einen Nachfolger besitzt} \\ 1 + Z(v') & \text{sonst} \end{cases} \tag{3.7}$$

wobei v' den Vorgänger des Knotens v darstellt. Die Zweigtiefe wird direkt nach der Erstellung des Knotens berechnet und daraufhin nicht verändert. Ein Beispiel für die Bestimmung der Zweigtiefe wird in Abbildung 3.3 gezeigt.

Der von Prusinkiewicz u.a. [PL04] beschriebene Begriff von axialen Bäumen [PL04, S.21] und die zugehörige Bestimmung der Achsentiefe unterscheidet sich von der Berechnung der Zweigtiefe. Knoten können dieselbe Zweigtiefe besitzen, selbst wenn ihre Positionen keine gerade Linie bilden.



(a) Ausgangssituation.

(b) Der Vorgänger des neuen Knotens besitzt genau einen Nachfolger.

(c) Die Zweigtiefe des rechten neuen Knotens ist um 1 größer als der seines Vorgängers, da dieser mehr als einen Nachfolger besitzt.

Abb. 3.3: Darstellung der Zweigtiefen von Knotenpunkten aus Abbildung 3.1.

Zusätzliche Bedingungen

Um mehr Kontrolle über die vom Space Colonization Algorithmus generierte Baumstruktur zu bekommen, wurden zusätzliche Bedingungen an den Ablauf des Algorithmus hinzugefügt:

***max*$_{grad}$** Der maximale Grad der Knotenpunkte im Baum. Bevor ein neuer Knotenpunkt v_n als Nachfolger eines Knotenpunkts v erstellt wird, wird die Anzahl der Nachfolger von v überprüft. Entspricht diese max_{grad}, wird der Knotenpunkt v_n nicht erstellt.

max_Z Die maximale Zweigtiefe eines Knotens. Bevor ein neuer Knotenpunkt v' als Nachfolger eines Knotenpunkts v erstellt wird, wird seine Zweigtiefe $Z(v')$ berechnet. Übersteigt diese max_Z, wird der Knotenpunkt v' nicht erstellt.

max_NG Die maximale Anzahl von Iterationen, in welchen einem Knotenpunkt kein neuer Nachfolger hinzugefügt wurde.[1] Hat ein Knoten diesen Wert erreicht, wird er in Schritt 1 des Algorithmus 3.3 nicht weiter darauf untersucht, ob er sich innerhalb des Einflussradius eines Einflusspunktes befindet. Der Wert von max_{NG} eines Knotenpunkts wird auf 0 zurückgesetzt, falls diesem ein neuer Nachfolger hinzugefügt wurde. Durch die vorsichtige Wahl von max_{NG} können Positionsvergleiche vermieden werden ohne den resultierenden Baum zu verändern.

Gewichtetes Wachstum

Gewichtetes Wachstum ist eine Möglichkeit die Schrittweite D in Abhängigkeit von der Zweigtiefe zu erhöhen. Befindet sich ein Knotenpunkt auf der Zweigtiefe 0, werden neu hinzugefügte Nachfolger im Abstand von $2 * D$ positioniert, befindet er sich auf der maximalen Zweigtiefe, werden neu hinzugefügte Nachfolger im Abstand von D positioniert. Zwischen diesen beiden Zweigtiefen wird die Schrittweite linear interpoliert.

Kurvenreduktion

Kurvenreduktion in Abhängigkeit des Abzweigungswinkels ermöglicht es die Baumstruktur mithilfe einer verringerten Datenmenge darzustellen. Besitzt ein Knoten v genau einen Nachfolger v_n und einen Vorgänger v', werden die zwei Richtungsvektoren zwischen v' und v sowie v und v_n berechnet:

$$\overrightarrow{R_1} = \frac{\overrightarrow{p_v} - \overrightarrow{p_{v'}}}{\|\overrightarrow{p_v} - \overrightarrow{p_{v'}}\|} \text{ und } \overrightarrow{R_2} = \frac{\overrightarrow{p_{v_n}} - \overrightarrow{p_v}}{\|\overrightarrow{p_{v_n}} - \overrightarrow{p_v}\|} \tag{3.8}$$

Übersteigt das Skalarprodukt $\langle \overrightarrow{R_1}, \overrightarrow{R_2} \rangle$ einen festgelegten Maximalwert max_K zwischen 0 und 1, wird v aus dem Baum entfernt. v_n wird zum Nachfolger von v' und v' zum Vorgänger von v_n.

[1] *NG* steht für „No Grow“ oder „Did not grow“, englisch für „kein Wachstum“.

4

Implementierung

Die Implementierung des Projekts lässt sich in vier Bereiche unterteilen: Die Baumrepräsentation enthält Daten, die von den L-System und Space Colonization Implementierungen generiert werden. Diese Daten werden an das Modellgenerierungssystem übergeben, welches die Modelldaten für eine grafische Darstellung in der Unreal Engine 4 produziert.

4.1 Baumrepräsentation

Sowohl L-Systeme als auch der Space Colonization Algorithmus generieren einen graphentheoretischen Baum, auf Grundlage dessen die Modellgenerierung durchgeführt wird. Die implementierte Baumrepräsentation kann daher von beiden Systemen verwendet werden und ermöglicht es, diese mit demselben Modellgenerierungssystem zu visualisieren.

Der Baum wird durch eine Datenklasse repräsentiert, jedes Objekt dieser Klasse beschreibt einen Knoten sowie die Kante, welche vom Vorgänger zu dem Knoten führt. Die Datenklasse bietet Zugriff auf die folgenden Informationen:

Vorgänger und Nachfolger: Mithilfe eines Verweises auf den Vorgänger und eine Liste der Nachfolger eines Knotens kann der Baum-Graph vollständig repräsentiert werden. Weiterhin ermöglicht dies die Implementierung einer Reihe von rekursiven Funktionen zur Anpassung von Modelldaten.

Modell-Daten: Kanten werden, wie in Abschnitt 4.4.4 beschrieben, mithilfe von Zylindern visualisiert. Um die Generierung von Modelldaten zu vereinfachen, bietet die Datenklasse Zugriff auf Start- und Endposition, Start- und Endradius, Start- und Endnormale sowie einen Rotationswinkel.
Weiterhin wird die Zweigtiefe des repräsentierten Knoten gespeichert.

Wachstums-Daten: Die Wachstums-Daten bestehen aus einer Wachstumsrichtung, einem Einfluss-Zähler und dem „Kein Wachstum“-Zähler (NG-Counter), welche für den Ablauf des Space Colonization Algorithmus benötigt werden.

Ein Objekt der Datenklasse kann als Astsegment eines biologischen Baumes angesehen werden und wird durch das Modellgenerierungssystem als solches vi-

sualisiert. Im Folgenden wird der Begriff „Zweigobjekt“ verwendet, um ein Objekt der Baumrepräsentationsklasse zu bezeichnen.

4.2 L-Systeme

Die Implementierung einer L-System-Baumstruktur wird durch einen Unreal-Akteur verwirklicht, der im Level platziert werden kann. Nach Start des Levels wird das angegebene Axiom anhand der Produktionsregeln abgeleitet und die sich ergebende Zeichenkette von der Turtle-Implementierung interpretiert.

4.2.1 Parameter

Dem L-System-Akteur werden die folgenden Parameter über die Editor-UI übergeben:

Anzahl der Ableitungen: Die Anzahl der Ableitungen in $\mathbb{N}^+$, die auf dem Axiom durchgeführt werden sollen.

Axiom: Das Axiom in Form einer Zeichenkette.

Konstanten: Eine Konstante besteht aus der Angabe eines Identifikationssymbols und eines Wertes in $\mathbb{R}$. Konstanten können im Axiom und in den Nachfolgern der Produktionen verwendet werden.

Produktionen: Jede Produktion besteht aus Angabe eines Vorgängers, eine Liste von Parametern und einem Nachfolger. Der Vorgänger und jeder Parameter entspricht einem einzelnen Symbol, der Nachfolger wird als Zeichenkette eingetragen. Die Parametersymbole können nur innerhalb des Nachfolgers verwendet werden. Der Vorgänger und die Liste der Parameter bilden das parametrische Wort, welches bei einer Ableitung durch den Nachfolger ersetzt wird.

Tropismus: Der Einfluss von Tropismus in Form eines dreidimensionalen Vektors und eines Biegsamkeitswertes.

Für das Axiom und die Produktionen gelten die in Kapitel 2 festgelegten Regeln für die Definition von L-Systemen. Weiterhin gelten die üblichen Regeln für die Angabe von arithmetischen Operationen. Die Verwendung von Klammern ist jedoch auf die Angabe von Parametern eines parametrischen Wortes beschränkt, ihre Verwendung zur Beeinflussung der Auswertungsreihenfolge eines arithmetischen Ausdrucks wird nicht unterstützt.

Die Parameter für eine Anpassung der Visualisierung der Turtle-Interpretationen werden in Abschnitt 4.4.2 erläutert.

Ein Beispiel für die korrekte Eingabe eines L-Systems über die Editor-UI wird in Abbildung 4.1 gezeigt.

Abb. 4.1: Ein Beispiel für die Angabe des L-Systems aus Gleichung 2.5b mit der resultierenden Baumstruktur aus Abbildung 2.5b.

4.2.2 Ableitung

Zu Anfang der Erstellung des L-System-Akteurs werden alle Konstantensymbole im Axiom und den Produktionen durch die Konstantenwerte ersetzt.

Die Implementierung arbeitet durchgehend auf derselben Zeichenkette, angefangen mit dem Axiom. In jeder Ableitung werden die in den Produktionsregeln definierten parametrischen Wörter durch die angegebenen Nachfolger ersetzt.

Nachdem die vorgegebene Anzahl von Ableitungen durchgeführt wurde, wird die resultierende Zeichenkette an die Turtle-Implementierung weiter gegeben.

4.2.3 Turtle Interpretation

Die Turtle-Implementierung bekommt die aus den Ableitungen resultierende Zeichenkette übergeben. Diese wird daraufhin sequentiell abgearbeitet und entsprechend den in Abschnitt 2.4 vorgestellten Konzepten interpretiert. Der resultierende Baum wird für die Konstruktion des Modells an das Modellgenerierungssystem weitergegeben.

4.3 Space-Colonization Algorithmus

Die Implementierung einer Space-Colonization-Baumstruktur stellt sich aus der Platzierung von mindestens einem Akteur für die Repräsentation des Einflussbereichs und einem Akteur für die Umsetzung des Algorithmus zusammen.

4.3.1 Parameter

Dem Space-Colonization-Akteur werden Parameter des ursprünglichen Algorithmus sowie die Eingaben für in Abschnitt 3.5 besprochene Erweiterungen über die Editor-UI übergeben. Zu den ursprünglichen Parametern gehören der Minimalradius, Einflussradius, Schrittweite, ein Tropismusvektor sowie die Anzahl der durchzuführenden Iterationen. Zu den erweiterten Parametern gehören der maximale Grad, die maximale Zweigtiefe, die maximale Anzahl von „Kein Wachstum"-Iterationen und eine Abfrage, ob gewichtetes Wachstum durchgeführt werden soll.

Weiterhin müssen die zugeordneten Einflussbereich-Akteure angegeben werden – damit der Algorithmus durchgeführt werden kann, muss mindestens einer dieser Akteure mit mindestens einem Einflusspunkt eingetragen werden.

4.3.2 Einflussbereiche

Durch die Platzierung von Einflussbereich-Akteuren kann die Verteilung der Einflusspunkte mithilfe des Unreal-Editors beeinflusst werden. Es sind derzeit zwei Formen von Einflussbereichen wählbar: Eine Kugel- und eine Zylinderform. Die Kugelform benötigt die Eingabe eines Kugelradius während die Zylinderform durch eine Höhe und einen Radius beschrieben wird.

Einem Space-Colonization-Akteur können mehrere Einflussbereiche zugeordnet werden, um eine bestimmte Baumstruktur zu formen. Jeder Einflussbereich-Akteur wird weiterhin in einem vorgegebenen Abstand von dem Space-Colonization-Akteur platziert.

Dem Einflussbereich muss eine positive Anzahl von zu generierenden Einflusspunkten und ein Random Seed Wert übergeben werden. Ein Random Seed ist ein Wert, der von einem Zufallsgenerator verwendet wird, um eine Folge von zufälligen Zahlen zu generieren. Bei Verwendung desselben Random Seed wird dieselbe Folge von Zufallszahlen erstellt, was eine Kontrolle über Generierung ermöglicht. Somit wird, wenn auch alle anderen Parameter übereinstimmen, mit demselben Random Seed Wert dieselbe Baumstruktur aufgebaut.

4.3.3 Ablauf des Algorithmus

Die Einflusspunkte aller dem Space-Colonization-Akteur zugeordneten Einflussbereiche werden diesem zu Beginn der Baum-Generierung übergeben. Daraufhin wird ein Baum entsprechend der in Abschnitt 3.4 und Abschnitt 3.5 vorgestellten Konzepten aufgebaut und für die Konstruktion des Modells an das Modellgenerierungssystem weitergegeben.

4.4 Modellgenerierung

Das Modellgenerierungssystem erhält einen graphentheoretischen Baum von L-System- und Space-Colonization-Akteuren und generiert ein dreidimensionales Mesh in der von der Unreal Engine geforderten Form. Das Mesh entspricht der

Visualisierung der Kanten des Baums in Form von Zylindern und simuliert dadurch vereinfacht die Aststruktur eines biologischen Baumes. [RLP07, Abschn. 2]

4.4.1 Procedural Mesh Component

Die Procedural Mesh Component ist eine Komponente der Unreal Engine, welche die Darstellung von prozedural generierten Polygonnetzen zulässt. Der Komponente werden Vertexdaten in Form von Listen aus Positions-, Normalen-, Tangenten-, Textur- und Indexdaten übergeben. Das Grafiksystem der Unreal Engine ist daraufhin in der Lage, die Komponente als dreidimensionales Modell im Level darzustellen. [Pro] Die Vertexdaten werden, basierend auf dem übergebenen Baum und den Parametern, vom Modellgenerierungssytem erstellt.

Jedem L-System-Akteur und Space-Colonization-Akteur ist eine Procedural Mesh Component zugeordnet.

4.4.2 Parameter

Dem Modellgenerierungssystem werden die folgenden Parameter über die Editor-UI übergeben:

Radius-Daten: Dies beinhaltet den Blattradius, den Radiuswachstumswert, den Stammbreitenmultiplikator und die Abfrage, ob Radiusberechnungen durchgeführt werden sollen.

Genauigkeit: Dies beinhaltet die minimale und maximale Anzahl von Zylindersektionen sowie den Kurvenreduktionswert.

Sonstiges: Weiterhin wird ein Startrotationswinkel, ein Material und eine Abfrage, ob ein fraktales Mesh erstellt werden soll, übergeben. Ein Material beinhaltet Textur- und Shaderinformationen und ist für die Oberflächenbeschaffenheit des generierten Modells verantwortlich.

4.4.3 Operationen auf dem Baum

Folgende Operationen werden vor Beginn der Modelldatengenerierung auf dem graphentheoretischen Baum durchgeführt:

Kurvenreduktion: Die Kurvenreduktion wird, beginnend mit dem Wurzel-Zweigobjekt des Baums, rekursiv ausgeführt. In jedem Schritt wird überprüft ob, entsprechend der Beschreibung in Paragraph 3.5, das aktuelle Zweigobjekt entfernt werden kann. Falls nicht, wird die Kurvenreduktion auf allen Nachfolgern des aktuellen Zweigobjekts durchgeführt. Die Rekursion bricht ab, falls ein Zweigobjekt keine Nachfolger besitzt.
Der Parameter „Kurvenreduktionswert“ entspricht dem Maximalwert des Skalarprodukts max_K.

Radiusberechnung: Der Endradius jedes Zweigobjekts wird, beginnend mit dem Wurzel-Zweigobjekt des Baums, rekursiv anhand von Gleichung 3.6 berechnet. Der Parameter „Blattradius“ entspricht r_0 und der „Radiuswachstumswert“ entspricht g.
Der Startradius jedes Zweigobjekts wird auf den Wert des Endradius seines Vorgängers gesetzt. Da das Wurzel-Zweigobjekt keinen Vorgänger besitzt, wird der Startradius aus der Multiplikation des Stammbreitenmultiplikator mit dem Endradius des Objekts bestimmt.

Normalenberechnung: Die Endnormale $\overrightarrow{n_e}$ jedes Zweigobjekts werden mithilfe der Startposition $\overrightarrow{p_s}$ und Endposition $\overrightarrow{p_e}$ sowie seiner Startnormale $\overrightarrow{n_s}$ wie folgt berechnet:

$$\overrightarrow{n_e} = \frac{(\overrightarrow{p_e} - \overrightarrow{p_s}) + \overrightarrow{n_s}}{\|(\overrightarrow{p_e} - \overrightarrow{p_s}) + \overrightarrow{n_s}\|} \tag{4.1}$$

Die Startnormale entspricht der Endnormale des Vorgängers, im Falle des Wurzel-Zweigobjekts entspricht die Startnormale $\overrightarrow{n_s} = \overrightarrow{p_e} - \overrightarrow{p_e}$.

Verringerung der Abzweigungswinkel: Die Nachfolger jedes Zweigobjekts werden, entsprechend der Beschreibung in Paragraph 3.4, einander angenähert. Die Verringerung der Abzweigungswinkel wird nur bei einem durch einen Space-Colonization-Akteur generierten Baum durchgeführt.

4.4.4 Generierung der Zylinder-Meshes

Moderne Grafik-APIs stellen dreidimensionale Modelle in Form von geometrischen Primitiven – in diesem Fall Dreiecke – dar. Je drei Vertizes – Punkte in der Welt – bilden ein Dreieck, dessen Oberfläche mithilfe eines übergebenen Materials gefärbt wird.

Die Vertexe bestehen, zusätzlich zu ihren Positionen, aus :

Normalenvektoren, welche die Richtung darstellen, von der aus das Dreieck sichtbar ist und für Beleuchtungsberechnungen benötigt werden. [Moda]

Tangentenvektoren, welche orthogonal zu den Normalenvektoren stehen und für erweiterte Beleuchtungsberechnungen benötigt werden.

Texturkoordinaten, welche für das korrekte Auftragen von Texturen auf das Modell benötigt werden. [Moda]

Der Mantel eines Zylinder kann durch die Verbindung von zwei Kreisen generiert werden.

Berechnung der Vertizes auf einem Kreis

Da ein Kreis theoretisch aus unendlich vielen Punkten besteht, muss eine gewisse Genauigkeit bei der Darstellung von runden Modellen festgelegt werden – ein Kreis wird aus einer festgelegten Anzahl von Segmenten generiert. Mithilfe eines

Mittelpunkts $\overrightarrow{c}$ und einem Radius r kann ein Kreis im zweidimensionalen Raum beschrieben werden. Die Vertexpositionen $\overrightarrow{v_i}$ werden wie folgt berechnet:

$$\overrightarrow{v_i} = \overrightarrow{c} + r * \begin{pmatrix} cos(d) \\ sin(d) \\ 0 \end{pmatrix} \text{ mit } d = rot_z + i * \frac{360°}{n} \text{ und } i = 0...(n-1) \tag{4.2}$$

[Modb]

wobei n der Anzahl der Segmente und r_z einem Startrotationswinkel entspricht.

Mithilfe der Kreisnormalen $\overrightarrow{n_k}$, die orthogonal zur Kreisebene steht und des Kreismittelpunkts $\overrightarrow{c}$ kann die Kreisebene beschrieben werden, auf welcher die Vertizes zu generieren sind. Eine Vertexposition $\overrightarrow{v_i}$ muss um den Kreismittelpunkt rotiert werden, um auf der Kreisebene zu liegen, wobei Rotationsachse $\overrightarrow{R}$ und Rotationswinkel α wie folgt berechnet werden:

$$\overrightarrow{R} = \frac{(\overrightarrow{v_i} - \overrightarrow{c}) \times \overrightarrow{n_k}}{\|(\overrightarrow{v_i} - \overrightarrow{c}) \times \overrightarrow{n_k}\|} \tag{4.3}$$

$$\alpha = arccos(\langle \overrightarrow{n_k}, \overrightarrow{z} \rangle) \text{ mit } \overrightarrow{z} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{4.4}$$

wobei *arccos* dem Arkuskosinus entspricht. [Rot] Die rotierte Vertexposition $\overrightarrow{v}$ ermöglicht die Berechnung der Vertexnormale $\overrightarrow{n_v}$ und Vertextangente $\overrightarrow{t_v}$:

$$\overrightarrow{n_v} = \frac{\overrightarrow{v} - \overrightarrow{c}}{\|\overrightarrow{v} - \overrightarrow{c}\|} \text{ und } \overrightarrow{t_v} = \overrightarrow{n_v} \times \overrightarrow{n_k} \tag{4.5}$$

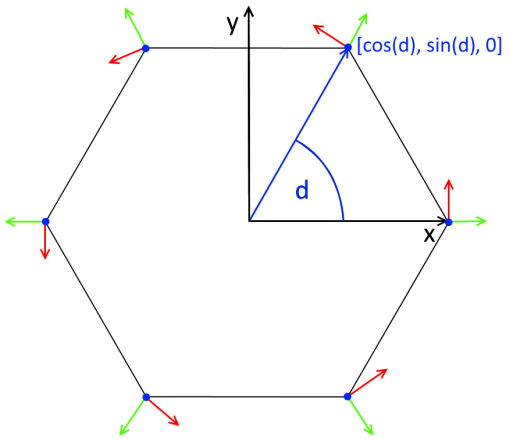
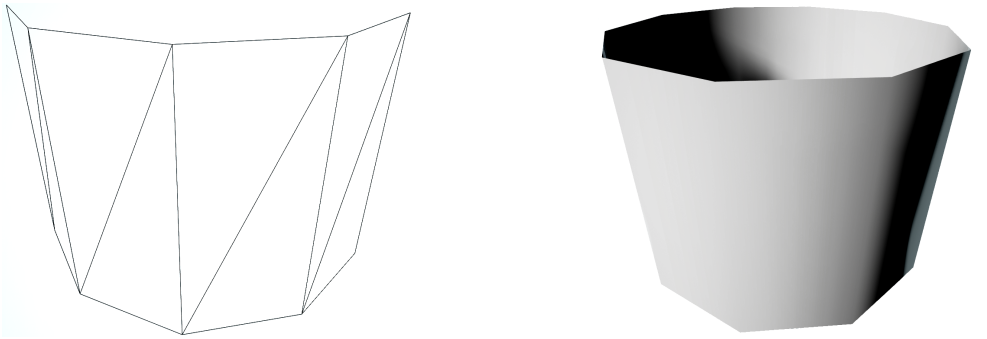


Abb. 4.2: Beispiel für die Berechnung von Vertexpositionen auf einem Einheitskreis mit einer Genauigkeit von sechs Segmenten. $d = \frac{360°}{6} = 60°$. Die blauen Punkte entsprechen den Positionen, die grünen Pfeile den Normalen und die roten Pfeile den Tangenten der Vertizes. Eigene Abbildung.

Verbindung der Kreise

Um den Zylindermantel eines Zweigobjekts zu bilden, werden zwei Kreise mithilfe der Start- und Enddaten des Objekts generiert. Jedes Kreissegment eines Kreises wird mit dem entsprechenden Segment des anderen Kreises verbunden und bildet dadurch ein Zylindersegment. Ein Zylindersegment entspricht somit einem Rechteck. Da das Modell jedoch Dreiecksdaten benötigt, wird jedes Rechteck, wie in Abbildung 4.3a dargestellt, aus zwei Dreiecken gebildet. [Modb]



(a) Darstellung der Dreiecke, welche bei der Verbindung der Kreise entstehen.

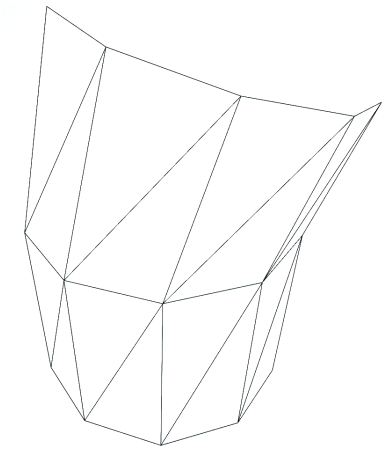
(b) Gefärbte Dreiecke mit Beleuchtungsberechnung.

Abb. 4.3: Verbindung zweier Kreise zu einem Zylindermantel. Eigene Abbildungen.

Wird jedes Zweigobjekt durch die Verbindung von genau zwei Kreisen dargestellt, führt dies zu der Generierung redundanter Vertexdaten. Die Enddaten eines Zweigobjekts und die Startdaten seines Nachfolgers entsprechen einander, die daraus generierten Kreis-Vertizes liegen genau aufeinander. Anstatt nun vier Kreise für die Generierung zweier Zylinder zu verwenden, können die Vertizes des verbindenden Kreises wiederverwendet werden – es genügen drei Kreise für die Generierung zweier Zylinder.

Die Verbindung der Modelldaten kann für alle Zweigobjekte durchgeführt werden, deren Nachfolger dieselbe Zweigtiefe besitzen und somit eine Folge von zusammenhängenden Zylindermodellen bilden. Für einen Nachfolger mit einer sich unterscheidenden Zweigtiefe wird eine neue Folge von zusammenhängenden Zylindermodellen begonnen. [Modb]

Ein Beispiel für die Generierung zweier Zylinder mithilfe von drei Kreisen wird in Abbildung 4.4a dargestellt.



(a) Darstellung der Dreiecke, welche bei der Verbindung der Kreise entstehen.

(b) Gefärbte Dreiecke mit Beleuchtungsberechnung.

Abb. 4.4: Verbindung dreier Kreise zu einer Folge zusammenhängender Zylindermodelle. Eigene Abbildungen.

5

Ergebnisse

In diesem Kapitel werden die Ergebnisse vorgestellt, die mithilfe den Implementierungen von L-Systemen und dem Space-Colonization Algorithmus produziert werden können.

5.1 L-Systeme

L-Systeme ermöglichen eine

5.1.1 Monopodial

test

5.1.2 Simpodial

test

5.1.3 Ternäre Verzweigungen

test

5.1.4 Tropismus

test

5.1.5 Performanz

test

5.2 Space-Colonization Algorithmus

test

5.2.1 Ursprüngliche Parameter

test

5.2.2 Erweiterte Parameter

test

5.2.3 Performanz

test

5.3 Probleme

Mit SCA: Gleicher Abstand von zwei Einflusspunkten führt zu beinahe unendlicher Hin- und Her-Generierung von Branches.

SCA: Random-Verteilung der Punkte.

6

Zusammenfassung und Ausblick

Zusammenfassung

6.1 Erweiterungen

test

6.1.1 Space Colonization Algorithmus

test

Positionsabfragen

Verbesserte Positionsvergleiche bei Einflusspunkt- zu Knotenpunkt-Abfragen.

Einflussbereiche

Verbesserte Möglichkeit, Einflussbereiche anzugeben. Oberfläche, Random-Verteilung

6.1.2 Aststruktur

Generalized cylinders.

6.1.3 Texturen

test

6.1.4 Blätter

test

6.1.5 Generierung zur Laufzeit

test

6.1.6 Verteilung

test

6.1.7 Instanziierung

Eine beschränkte Anzahl Pflanzen generieren und diese so verteilen, dass es nicht bemerkt wird, dass es immer die selben Pflanzen sind

6.1.8 title

test

Literaturverzeichnis

DL05. Deussen, Oliver und Bernd Lintermann: *Digital Design of Nature - Computer Generated Plants and Organics*. Springer-Verlag Berlin Heidelberg 2005, 2005.

GSJ04. Goldman, Ron, Scott Schaefer und Tao Ju: *Turtle Geometry in Computer Graphics and Computer Aided Design*, 2004. http://www.cs.wustl.edu/~taoju/research/TurtlesforCADRevised.pdf.

Lux14. Lux, Prof. Dr. Andreas: *Algorithmen und Datenstrukturen - Vorlesungsskript Kapitel 4*, 2014.

Man83. Mandelbrot, Benoit B.: *The Fractal Geometry of Nature*. W. H. Freeman and Company, 1983.

Moda. *Modelling by numbers: Part One A: An introduction to procedural geometry*. http://www.gamasutra.com/blogs/JayelindaSuridge/20130903/199457/Modelling_by_numbers_Part_One_A.php.

Modb. *Modelling by numbers: Part Two A: The cylinder*. http://www.gamasutra.com/blogs/JayelindaSuridge/20130905/199626/Modelling_by_numbers_Part_Two_A.php.

PL04. Prusinkiewicz, Przemyslaw und Aristid Lindenmayer: *The Algorithmic Beauty of Plants*. Przemyslaw Prusinkiewicz, eBook Auflage, 2004. http://algorithmicbotany.org/papers/abop/abop.pdf.

Pro. *Procedural Mesh Component in C++:Getting Started*. https://wiki.unrealengine.com/Procedural_Mesh_Component_in_C%2B%2B:Getting_Started.

RFL+05. Runions, Adam, Martin Fuhrer, Brendan Lane, Pavol Federl, Anne-Gaëlle Rolland-Lagan und Przemyslaw Prusinkiewicz: *Modeling and visualization of leaf venation patterns*, 2005. http://algorithmicbotany.org/papers/venation.sig2005.pdf.

RLP07. Runions, Adam, Brendan Lane und Przemyslaw Prusinkiewicz: *Modeling Trees with a Space Colonization Algorithm*, 2007. http://algorithmicbotany.org/papers/colonization.egwnp2007.pdf.

Rot. *Maths - Angle between vectors*. http://www.euclideanspace.com/maths/algebra/vectors/angleBetween/index.htm.

Sch14. Schmitz, Prof. Dr. Heinz: *Theoretische Informatik - Vorlesungsskript*, 2014.

STN16. Shaker, Noor, Julian Togelius und Mark J. Nelson: *Procedural Content Generation in Games*. Springer International Publishing Switzerland 2016, 2016.

A

Glossar

Mesh	Englisch für Polygonnetz.

B

Erklärung der Kandidatin / des Kandidaten

☐ Die Arbeit habe ich selbstständig verfasst und keine anderen als die angegebenen Quellen- und Hilfsmittel verwendet.

☐ Die Arbeit wurde als Gruppenarbeit angefertigt. Meine eigene Leistung ist ...

Diesen Teil habe ich selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Namen der Mitverfasser: ...

Datum	Unterschrift der Kandidatin / des Kandidaten